Two-Level Address Storage and Address Prediction

Enric Morancho, José María Llabería and Àngel Olivé

Computer Architecture Department - Universitat Politècnica de Catalunya (Spain)[1]

Abstract. : The amount of information recorded in the prediction tables of the address predictors turns out to be comparable to current on-chip cache sizes. To reduce their area cost, we consider the spatial-locality property of memory references. We propose to split the addresses in two parts (high-order bits and low-order bits) and record them in different tables. This organization allows to record only once every unique high-order bits. We use it in a last-address predictor and our evaluations show that it produces significant area-cost reductions (28%-60%) without performance decreases.

1 Introduction

True-data and control dependencies are the major bottleneck for exploiting ILP. Some works [4][6][11] propose the use of prediction and speculation to overcome data dependencies. In load instructions, there is a true-data dependence between the address computation and the memory access; this dependence contributes to the large latency of the load instructions and can affect processor performance. Then, address predictors are valuable to access memory speculatively [4][10].

A typical address-prediction model is the last-address [11]. It assumes that a load instruction will compute the same address that the one computed in its previous execution. Proposals of last-address predictors [4][6] employ a direct-mapped Address Table (AT), indexed with some bits of the PC. Each AT entry contains the last address computed by a load instruction, a two-bit confidence counter, and a tag. We name this predictor *Base Last-Address Predictor* (BP).

Last address predictors use prediction tables that record up to 4.096 64-bit addresses [2][10], that is, 32 Kbytes; that is comparable to current on-chip cache sizes. However, the previous designs do not exploit the locality of the addresses. This property has been used in other works to take different advantages [3][12][13] (a detailed description of related works can be found in [9]). In this paper, we propose a new organization of the prediction table and we apply it to a typical last-address predictor to obtain a significant area-cost reduction.

This paper is organized as follows. Section 2 presents our proposal. Section 3 evaluates our proposal, and Section 4 summarizes the conclusions of this work.

---

1. *Author's address:* Computer Architecture Department, Universitat Politècnica de Catalunya, Jordi Girona 1-3, 08034 Barcelona (Spain). *E-mail:* enricm@ac.upc.es

A. Bode et al. (Eds.): Euro-Par 2000, LNCS 1900, pp. 960-964, 2000.

2 Two-Level Address Predictor

2.1 Basic Idea

Effective addresses exhibit temporal and spatial locality; it produces redundancy in AT contents. For instance, different accesses to the same global variable produce temporal redundancy. Also, variables stored in consecutive addresses and stack accesses produce spatial redundancy. We propose an organization to record them non redundantly.

AT is split in two parts: the Low-Address Table (LAT) and the High-Address Table (HAT). LAT records the low-order bits of the addresses and HAT the high-order bits; moreover, each LAT entry is linked to a HAT entry. Then, a HAT entry can be shared by several LAT entries. We apply this organization to the BP and we obtain the *Two-Level Address Predictor* (2LAP); Figure 1 shows its scheme.

To predict a load instruction, 2LAP indexes LAT using the load-instruction PC; this access obtains the low-order address portion and a link to HAT. After that, HAT is accessed to obtain the high-order portion. This sequential access does not imply an implementation restriction because LAT can be accessed early in the pipeline and the large number of pipeline stages before issuing an instruction; for instance, 5 stages in an Alpha 21264 [1]. Moreover, we could reduce the critical path of the speculative access by recording in LAT enough bits for indexing the cache.

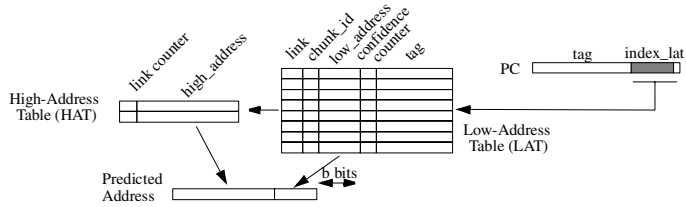

Fig. 1. Two-Level Address Predictor.

2.2 Locality Analysis and HAT Size

There is a trade-off between the number of HAT entries and the bits of the low-order portions (*b*). To obtain recommended sizes, we evaluate the locality in AT contents [9]; the suggestion is *b*=10, 12 or 14 and 64 HAT entries. We choose this HAT size because up to 96-entry HAT's can be accessed fully associatively in a single processor cycle [5].

2.3 Prediction-Table Management

2LAP updates LAT like BP updates AT, using the *always-allocate* policy. Also, before recording a high-order portion in HAT, 2LAP must verify if it is yet recorded. To check it, 2LAP looks for the high-order portion in HAT. If it is found (HAT hit), both entries are linked; if not, a HAT entry is evicted.

HAT Replacement and Tracing Empty HAT Entries

To reduce the eviction from HAT of useful information, we trace HAT entries not related to any LAT entry (empty HAT entries). To detect it, we relate to every HAT entry a link counter that reflects the number of LAT entries linked to it (*link_counter* in Figure 1). If none empty HAT entry is found, the replacement algorithm selects

randomly one HAT entry but the MRU one (no-MRU replacement algorithm). We have used this algorithm because the implementation of LRU algorithm is complex and expensive for large tables. We have evaluated the performance decrease produced by this decision: it is limited by 2.4% for *b*=10, 1% for *b*=12, and negligible for *b*=14.

2LAP updates the link counter on LAT replacements and on changes of the high-order portion related to a LAT entry. On HAT replacements, the LAT entries linked to the evicted HAT entry are not invalidated to simplify the design. This decision is possible because 2LAP is related to an speculative mechanism, but it produces mispredictions. Our experiments show that using three-bit link counters the performance of the 2LAP is almost saturated. These counters estimate empty HAT entries with a high correctness.

Filtering-Out Some High-Order Portions in HAT

2LAP allocates unpredictable load instructions in LAT but avoids the allocation in HAT of their high-order portions, i.e., their LAT entries are not linked to any HAT entry. Moreover, the link is broken when the classification of a load instruction changes from predictable to unpredictable, and only is re-established when it changes again.

The address chunk stored in *low_address* field of LAT is used to keep on updating the classification of the unpredictable instructions; the basic idea of this classification has been proposed in [7]. Also, the chunk is selected dynamically (*chunk_id* field in Figure 1) because the accuracy of 2LAP in programs with large-strided references (*ijpeg*) can be affected.

Filtering HAT Allocations and Managing Empty HAT Entries

We have evaluated the four possibilities that appear considering: a) filtering-out HAT allocations, and b) managing empty HAT entries. Our experiments show that both policies should be applied at the same time, specially in codes with a large working set of high order portions and medium-predictable instructions (*gcc*, *go*, *vortex*).

3 Evaluation: 2LAP versus BP

This section compares 2LAP versus BP, both using bounded prediction tables. Working-set size of static load instructions of the programs [8] justifies that the selected LAT sizes range from 256 to 4.096 entries.

3.1 Area Cost of the Predictors

We evaluate the area cost of a predictor as the amount of information that it records. Following formulas show the area cost of BP and 2LAP using 64-bit logical addresses, *t*-bit tags, 3-bit link counters, and *b*-bit address chunks.

$$\underset{AreaCost}{BasePredictor} = (t + 64 + 2) \times ATentries$$

$$\underset{AreaCost}{2LAP} = (3 + (64 - b)) \times HATentries + \left(\log_2 HATentries + \left\lceil \log_2 \frac{64}{b} \right\rceil + b + 2 + t\right) \times LATentries + \log_2 HATentries$$

Tag length influences on predictor accuracy. In the analysed codes, [8] shows that BP accuracy saturates when the number of index and tag bits is 17; then, we compare these configurations. The area-cost reduction from a BP to a 2LAP with the same number of AT and LAT entries ranges from 37% (256 entries, *b*=14) to 60% (4.096 entries, *b*=10).

3.2 Captured Address Predictability

The captured address predictability is defined as the percentage of correct predictions out of the number of executed load instructions. We will evaluate the predictability captured by several predictor configurations in the integer codes of the SPEC-95 with the largest working-set size of static load instructions (remaining programs present a similar behaviour but in a different table-size range). Our results were obtained running Alpha binaries instrumented with ATOM; programs were run until completion using reference input sets.

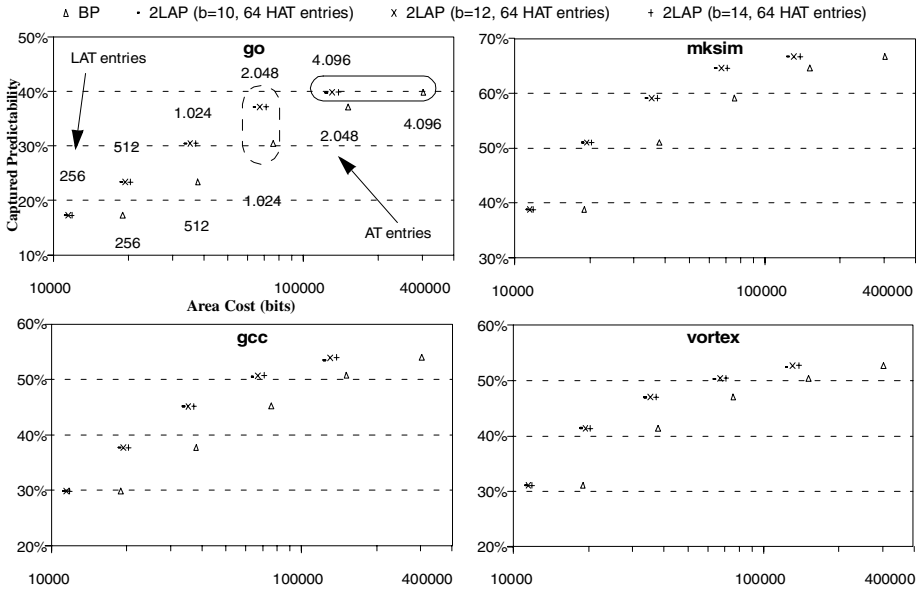

Fig. 2. Predictability captured by BP and 2LAP in several benchmarks. Horizontal axes stand for base-10 logarithm of predictor area cost, vertical axes stand for captured predictability.

Figure 2 shows the predictability captured by 2LAP and BP. Horizontal axes stand for area cost and vertical axes for captured predictability. Leftmost top graph is labelled with the number of AT and LAT entries of the configurations.Area-cost reduction from a 2LAP to a BP with the same number of LAT and AT entries do not represent a performance loss; for AT entries=LAT entries=4.096, a continuous oval surrounds these configurations. When LAT entries=2×AT entries (configurations surrounded by a dashed oval for LAT entries=2×AT entries=2.048), we obtain configurations with similar area cost. 2LAP outperform BP because LAT has less capacity misses than AT.

3.3 Accuracy

The accuracy of a predictor is defined as the percent of correct predictions out of the number of predictions. As every misprediction could produce a penalty of several processor cycles, the 2LAP should not present a lower accuracy than the BP. Our evaluations show that for b=10, 2LAP presents a slightly lower accuracy than the BP (in the worst benchmark -*gcc*- the difference is limited by 0.7%). For b=14, the difference is negligible. We present in [9] a detailed accuracy comparison.

4 Conclusions

We have shown that the spatial-locality property of the memory references produces redundancy in the prediction tables. We have taken advantage of this fact to reduce the area cost of the prediction tables. Our proposal splits the addresses computed by the load instructions in two parts: high-order and low-order portion. Addresses with the same high-order portion are recorded sharing one copy of the portion. Also, management of empty HAT entries, and filtering-out allocations of high-order portions related to unpredictable instructions improve the performance of the proposal.

Other prediction models (stride, context and hybrid) can also take advantage of the locality of the addresses to reduce their area cost.

This work proposes a new organization of the prediction table but it maintains the allocation policy, then, our proposal predicts the same instructions than the BP, and IPC speed-up is the one reported in other works [2][4][10].

Acknowledgements

This work was supported by the spanish government (grant CICYT TIC98 0511-C02-01), and the CEPBA (European Centre for Parallelism of Barcelona)

References

1. Alpha 21264 MicroProcessor Data Sheet. (1999). Compaq Computer Corporation.
2. B. Black, B. Mueller, S. Postal, R. Rakvic, N. Utamaphethai and J.P. Shen. (1998). Load Execution Latency Reduction. In *ICS-12*, pp. 29-36
3. M. Farrens and A. Park. (1991). Dynamic Base Register Caching: A Technique for Reducing Address Bus Width. In *ISCA-18*, pp. 128-137.
4. J. González and A. González. (1997). Speculative Execution via Address Prediction and Data Prefetching. In *ICS-11*.
5. B. Jacob and T. Mudge. (1998). Virtual Memory in Contemporary Microprocessors. IEEE Micro, Vol 18(4), pp. 60-75.
6. M.H. Lipasti, C. B. Wilkerson and J.P. Shen. (1996). Value Locality and Load Value Prediction. In *ASPLOS-7*.
7. E. Morancho, J.M. Llabería and À. Olivé. (1998). Split Last Address Predictor. In *PACT'98*.
8. E. Morancho, J.M. Llabería and À. Olivé. (1999). *Looking at History to Filter Allocations in Prediction Tables*. In *PACT99*, pp. 314-319
9. E.Morancho, J.M. Llabería and À. Olivé. (1999). Two Level Address Storage and Address Prediction. Technical report UPC-DAC-99/48.
10. G. Reinman and B. Calder. (1998). Predictive Techniques for Aggresive Load Speculation. In *MICRO-31*.
11. Y. Sazeides and J.E. Smith. (1996). The Predictability of Data Values. In *MICRO-29*.
12. A. Seznec. (1994). Decoupled sectored caches: reconciliating low tag volume and low miss rate. In *ISCA-21*.
13. H. Wang, T. Sung and Q. Yang. (1997). Minimizing Area Cost of On-Chip Cache Memories by Caching Address Tags. *IEEE Transactions on Computers*, 46 (11).